Part 1

1.1 Design decisions and reasoning

Filling in the skeleton and then training the models, I extract n-grams by padding the beginning of each text sequence and forming tuples of (context, character) pairs. Simple enough following our instructions and examples. I then calculate probabilities using maximum likelihood estimation, but here had to go ahead and implement Laplace smoothing to avoid zero probabilities for unseen characters. If we bumped into anything in the test data that was not in the training, I assigned equal probabilities to all characters to try avoiding undefined cases. All input was rendered lowercase for consistency, and the context was updated after each generated output character for the sake of coherence.

I had the flu while working on this assignment and was in something of a frustrating fugue state while debugging this - and more frustratingly, the GPT2 portion later. I *did* eventually find a set of small modifications to make to the perplexity function and my probability calculations that yielded results I was happy with, however. The issue was that my perplexities kept outputting as “Inf”, errors, or identical numbers. I briefly explored backoff strategies and setting non-zero default probabilities based on a Stack Overflow rant and a ChatGPT conversation respectively, but I was not happy with either of those solutions. The smoothing, in the end, is what gave me the most useful result.

1.2 Observations, and the “Monkeys with Typewriters” test…

Okay, starting out with the 2-gram output:

```
[>] Generated text using 2-gram model:
[_]
Fir inight full th by dione folike my mak, and brielle, with.

RON:
Nottake you he you; an st, ghtyierse our ne isderve gin thed,
Thome, my hince mus,--
The me par
wory teret, so sigue theres: 'tworfugus and my lor'?

Sookelis eny shearn racesine can
```

Pretty much absolute gibberish. Some actor indication structure, a colon followed by a potential quote… there is a semblance of structure at only the lowest level.

```
[>] Generated text using 3-gram model:
[_]
First plain'd not guilt loves. Bully
```

```
tal fore a come hous heir board anst she cound
dizzle was nurns fare a pill tonger posity you ans but the fath a
Fals of your nature,
And to sea guard Caesar be Beting that?
Or keep tendurse: commise
That yet mean
```

Like ye old stroke patient, this one is far more resembling of Shakespearean blathering but still lacks much coherence. Still many gibberish words.

```
[>] Generated text using 4-gram model:
[_]
First Gentlewomansinance:
I cross?

DUKE OF YORK:
What I knock,
Our parlike you hadst Citizen:
Neighbour had love pite;
Who's laught, which thee a worm,
The not folly, may place and queen upon case Aeneath suck'd
puissanio, like God's him the king me
```

While still nonsense, we're getting closer and closer to respecting the structure of the input text. Strange words still crop up, but the quality is notably improved and I'm not entirely convinced any of them are explicitly gibberish any longer.

```
[>] Generated text using 7-gram model:
[_]
First Citizen:
So may it be, sir,
That sweet sons and devout
By testament halting season'd and skittish for all that shall we have
surges, cracking thus long-usurped royal king!

KING HENRY VI:
The hand
Hath newly knit,
And shake
Your speed your love
```

The 7-gram model is very impressive. I read this and feel like I am sleepy and being spoken to by a real, albeit hopelessly flowery, individual. We start to see the text generation that makes people look at AI output and wonder if there is a spark of intelligence or meaning - very cool!

ALL of these outputs are better than monkeys on typewriters, and it really didn't take long to get *significantly* better than said monkeys. I really enjoyed changing my random seed for the model and parsing the 4-gram and 7-gram results, and there were some real gems - though I've opted to go with my best run for this report.

1.3 Perplexity metrics

```
[>] Computing perplexity on NYTimes Article...
[>] 2-gram model average perplexity: 11.07
[>] 3-gram model average perplexity: 9.70
[>] 4-gram model average perplexity: 10.45
[>] 7-gram model average perplexity: 30.36

[>] Computing perplexity on Shakespeare Sonnets...
[>] 2-gram model average perplexity: 6.91
[>] 3-gram model average perplexity: 5.09
[>] 4-gram model average perplexity: 5.09
[>] 7-gram model average perplexity: 12.29
```

My calculated perplexity metrics, for the most part, make sense. As could be predicted, the Shakespearean N-gram models' outputs are not too close to the modern English of the New York Times. The Sonnets had much lower scores (indicating a closer match of course), though interestingly the 7-gram was considered farther off?

1.4 Perplexity thoughts

As for what to make of this, I think the issue was that the 7-gram model overfitted and failed to generalize. It memorized sequences, sure, but really could not capture the more varied nature of the text corpora. I really find it weird that the 7-gram on the Sonnets did worse than the majority of the NYT samples… this does track though, as additional runs really had me thinking that the 4-gram model was more "interesting", as subjective a metric as that really is.

In any case, the fact that the models generally performed worse on the NYT text is to be expected, and indicates that my altered perplexity function was still doing its job more or less. I'd say that these results highlight the limitations of n-gram models, particularly their strong dependence on domain-specific data and lack of grounds for… context-awareness of a higher order? I'm not sure how to phrase the mismatch between the generated outputs and the more natural language of, say, GPT. But that's where part two comes in!

Part 2

2.1 Experimental settings

I went ahead and fine-tuned my GPT2 model at the character level by tweaking the tokenizer to accept individual characters (letters/punctuation/whitespace/newlines/etc.). I then trained it on

the same dataset as the earlier n-gram models using the Hugging Face Trainer, which was a first for me. I padded things with a “[PAD]” sequence and used a max sequence length of 128 for training, and after far too much trial and error settled on the hyperparameters below.

Training Settings:

- Epochs: 1
    - I wish I’d tried for more once I finally did enough troubleshooting to lower the time training took… I was not happy with my output.
- Batch Size: 64 (with gradient accumulation of 8)
- Learning Rate: 3e-4
- Weight Decay: 0.01
- Mixed Precision: Enabled (fp16=True)
- Trained on my NVIDIA RTX 3080 GPU for about 17ish minutes, though this varied a lot between runs.

Generation Settings:

- Used model.generate with do_sample=True.
- Temperature: 0.7 (slightly more deterministic - things REALLY fell apart with higher values).
- Top-k: 50
- Top-p: 0.9
- Max Length: 250 tokens

The largest issue I encountered with this part of the assignment was just incessant Out Of Memory (OOM) errors on the Google Colab. I’d fiddle with a few settings, and things would grind to a halt. I found out how to dynamically adjust my training parameters, and that did eventually help to an extent, but overall this was a tedious process. Being sick for a week and then hustling to get part 2 done was a real ordeal.

In the end, I opted to port things over to my personal machine, and the training times sped up *significantly* across my parallelized cards. This took a while to set up and troubleshoot compatibility issues with CUDA, but was worth it for the sake of iteration and exploration.

2.2 Comparing outputs!

Sample Outputs:

```
Wherefore art the bark and the time of my heart. i will not it?
Wherefore art i will not as the love of her; i will not her;
Whomst thou lovest thy good fortune to the sea; and thou
Whomst i pray you, and shall be company, for the weight
Thou art a part of person, what say you, god so? god so!
Thou art the grace of your daughter loves. what you have?
Alas, poor passion, sir, the mother's strong, and they is a
Alas, poor and converate for the sight, she is the death.
Dost thou art a more was to a such a garlant? no, no. no, i
```

```
Dost thou art more to and before than o'er the power. ho! i
```

Interestingly, despite expecting GPT-2 to outperform n-grams, I actually found my 4-gram and 7-gram models often produced text that felt more traditionally “Shakespearean” (e.g., *“So may it be, sir, That sweet sons and devout…”*) than GPT-2’s somewhat disjointed lines (like *“Wherefore art the bark and the time of my heart. i will not it?”*).

This may honestly just be due to the higher order N-grams just memorizing extended sequences, yielding coherent segments in the same style as the training text. The bones are here, the structure, but I just do not see the same quality subjectively. No, not even objectively, honestly…

GPT2 showed glimpses of longer context-awareness (pronouns consistency, subject agreement, etc), but with only 1 epoch of training, it mostly just produced awkward or incomplete sentences. I’m unable to remote into the machine I ran this on and retrain with a higher number of epochs because today’s thunderstorm in Oakland knocked it out, but I will retrain and update this section if I can before submission with a GPT2 model trained with three epochs. I’m hoping it makes a considerable difference.

In short, my limited GPT-2 fine-tuning didn’t consistently surpass the best n-gram results. I definitely want to keep tweaking this model when time permits.